

Anatomy of a Screenshot

Rodrigo Chiossi

Background (Or where did this talk came from!)

- Project requirements:
 - Implement a new SystemUI feature
 - Take screenshot, process, present the result to the user.
- Constraints
 - Results must be generated within 500ms
- Target
 - Android 4.4.4

Objectives

- Starting from the event trigger (button event detected) get to OpenGL calls.
- Analyse filtering performance.
- Emulator problems and fixes.
- Improvements in Android 5.0+

PS: All code snippets in this presentation are from android tag android-4.4.4_r1

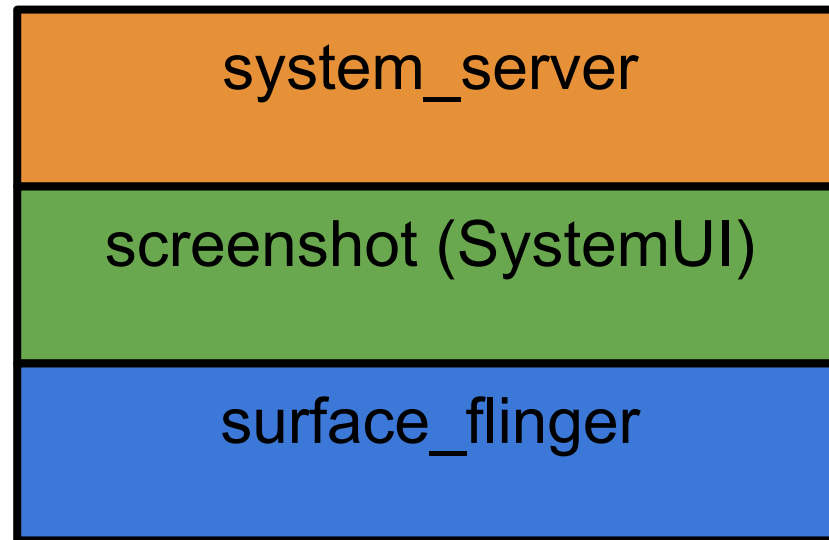
Screenshot: Who is involved

Considering the usual method of taking a screenshot in an Android Phone, three processes take part:

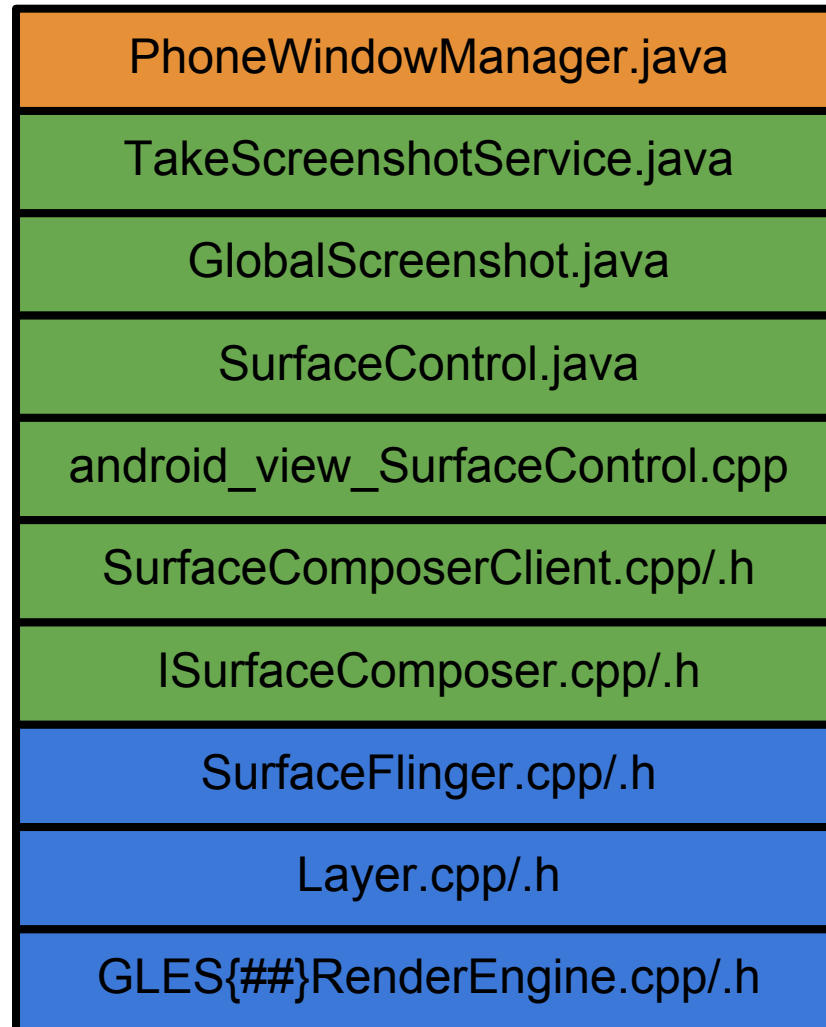
- system_server
- screenshot (SystemUI)
 - `android:process=":screenshot"`
- surface_flinger

PS: Usual Method - Power Button + Volume Down Button

Process Stack



Source File Stack



First: frameworks/base

PhoneWindowManager.java

- Intercept ScreenshotChord()
 - Button trigger event is identified
 - mScreenshotRunnable posted
 - Start and bind to TakeScreenshot service

```
3753         ComponentName cn = new ComponentName ("com.android.systemui",
3754                                             "com.android.systemui.screenshot.TakeScreenshotService" );
3755         Intent intent = new Intent ();
...
3794         if (mContext.bindServiceAsUser (
3795             intent, conn, Context.BIND_AUTO_CREATE, UserHandle.CURRENT)) {
```


TakeScreenshotService.java

- Instantiate GlobalScreenshot
- call takeScreenshot()

```
38         if (mScreenshot == null) {  
39             mScreenshot = new GlobalScreenshot (TakeScreenshotService.this);  
40         }  
41         mScreenshot.takeScreenshot (new Runnable () {
```

GlobalScreenshot.java

- Check display metrics and rotation
- Take the screenshot
- Generate the final bitmap
 - Apply rotation if required
- Show Screenshot animation

```
438     mDisplay.getRealMetrics (mDisplayMetrics);
439     float[] dims = {mDisplayMetrics.widthPixels, mDisplayMetrics.heightPixels};
...
451     // Take the screenshot
452     mScreenBitmap = SurfaceControl.screenshot ((int) dims[0], (int) dims[1]);
```

SurfaceControl.java

- Set the layers to be included in the screenshot.
 - In this case, all layers are included.
- Move to native code.

```
605     public static Bitmap screenshot(int width, int height) {  
    ...  
607         IBinder displayToken = SurfaceControl.getBuiltInDisplay(  
608             SurfaceControl.BUILT_IN_DISPLAY_ID_MAIN);  
609         return nativeScreenshot(displayToken, width, height, 0, 0, true);  
610     }
```

PS: Layers

- Defines the type of the Window and it's Z order.
- Defined in WindowManager.java
- Managed by PhoneWindowManager.java
 - Assign an integer value to each layer
 - `windowTypeToLayer()`
- Calculated per WindowState in WindowManagerService
 - `assignLayersLocked()`

android_view_SurfaceControl.cpp

- Map java methods to native methods.
 - nativeScreenshot() → nativeScreenshotBitmap()
- Creates a ScreenshotClient instance.
- Take the screenshot.
 - Call ScreenshotClient->update()
- Creates a bitmap for the screenshot.
 - Set pixel format.

```
126 ScreenshotClient* screenshot = new ScreenshotClient ();
127 status_t res = (width > 0 && height > 0)
128     ? (allLayers
129         ? screenshot->update(displayToken, width, height)
130         : screenshot->update(displayToken, width, height, minLayer, maxLayer))
131     : screenshot->update(displayToken);
```

Second: frameworks/native

SurfaceComposerClient.cpp/.h

- Defines ScreenshotClient.
- Gets the Interface to SurfaceComposer.
- Capture the screen.
 - ISurfaceComposer->captureScreen()

```
654     sp<ISurfaceComposer> s(ComposerService::getComposerService());  
...  
664     status_t err = s->captureScreen(display, mBufferQueue,  
665                                   reqWidth, reqHeight, minLayerZ, maxLayerZ);
```

ISurfaceComposer.cpp/.h

- Defines BnSurfaceComposer class.
 - “Implements” ISurfaceComposer interface.
- Send CAPTURE_SCREEN message.
- Call captureScreen() when message is received on target.

```
118         remote()->transact(BnSurfaceComposer::CAPTURE_SCREEN, data, &reply);  
...  
279         case CAPTURE_SCREEN: {  
...  
288             status_t res = captureScreen(display, producer,  
289                 reqWidth, reqHeight, minLayerZ, maxLayerZ);
```


SurfaceFlinger.cpp/.h

- Inherits BnSurfaceComposer.
- Creates MessageCaptureScreen.
 - postMessageAsync().

```
2862     class MessageCaptureScreen : public MessageBase {  
    ...  
2884         virtual bool handler() {  
    ...  
2887             result = flinger->captureScreenImplLocked(hw,  
2888                 producer, reqWidth, reqHeight, minLayerZ, maxLayerZ);  
    ...  
2908             sp<MessageBase> msg = new MessageCaptureScreen (this,  
2909                 display, IGraphicBufferProducer::asInterface ( wrapper ),  
2910                 reqWidth, reqHeight, minLayerZ, maxLayerZ);  
2911  
2912             status_t res = postMessageAsync(msg);
```

SurfaceFlinger.cpp/.h

- `captureScreenImplLocked()`.
 - Calls `renderScreenImplLocked()`.
- `renderScreenImplLocked()`.
 - Set Viewport and Projection.
 - Iterate layers that will be drawn.

```
3023         renderScreenImplLocked(hw, reqWidth, reqHeight,  
3024                               minLayerZ, maxLayerZ, true);  
...  
2938     engine.setViewportAndProjection (reqWidth, reqHeight, hw_w, hw_h, yswap);  
...  
2946     for (size_t i=0 ; i<count ; ++i) {  
...  
2953         layer->draw(hw);
```

Layer.cpp/.h

- draw() → onDraw() → drawWithOpenGL()
- Get RenderEngine
 - OpenGL ES 1.0, 1.1 or 2.0
- drawMesh()

```
598     RenderEngine & engine (mFlinger->getRenderEngine ());  
...  
600     engine.drawMesh (mMesh);
```

GLS{##}RenderEngine.cpp/.h

- Different versions for OpenGL ES 1.0, 1.1 and 2.0.
- Implements the actual drawing.

```
186 void GLS20RenderEngine::drawMesh(const Mesh& mesh) {  
    ...  
205     glDrawArrays(mesh.getPrimitive(), 0, mesh.getVertexCount());  
}
```

Timing

- Time to take a screenshot and show the result screen inside an ImageView
 - \sim 600ms
- Too high for smooth user experience
- Could it be optimized?

Filtering

```
2920 void SurfaceFlinger::renderScreenImplLocked(...)
2925 {
2929     // get screen geometry
2930     const uint32_t hw_w = hw->getWidth();
2931     const uint32_t hw_h = hw->getHeight();
2932     const bool filtering = reqWidth != hw_w || reqWidth != hw_h;
2933
2944     const LayerVector& layers( mDrawingState.layersSortedByZ );
2945     const size_t count = layers.size();
2946     for (size_t i=0 ; i<count ; ++i) {
2947         const sp<Layer>& layer(layers[i]);
2948         const Layer::State& state(layer->getDrawingState());
2949         if (state.layerStack == hw->getLayerStack()) {
2950             if (state.z >= minLayerZ && state.z <= maxLayerZ) {
2951                 if (layer->isVisible()) {
2952                     if (filtering) layer->setFiltering(true);
2953                     layer->draw(hw);
2954                     if (filtering) layer->setFiltering(false);
2955                 }
2956             }
2957         }
2958     }
```

Filtering Bug

```
2920 void SurfaceFlinger::renderScreenImplLocked(...)
2925 {
2929     // get screen geometry
2930     const uint32_t hw_w = hw->getWidth();
2931     const uint32_t hw_h = hw->getHeight();
- 2932     const bool filtering = reqWidth != hw_w || reqWidth != hw_h;
+ 2932     const bool filtering = reqWidth != hw_w || reqHeight != hw_h;
2933
2944     const LayerVector& layers( mDrawingState.layersSortedByZ );
2945     const size_t count = layers.size();
2946     for (size_t i=0 ; i<count ; ++i) {
2947         const sp<Layer>& layer(layers[i]);
2948         const Layer::State& state(layer->getDrawingState());
2949         if (state.layerStack == hw->getLayerStack()) {
2950             if (state.z >= minLayerZ && state.z <= maxLayerZ) {
2951                 if (layer->isVisible()) {
2952                     if (filtering) layer->setFiltering( true);
2953                     layer->draw(hw);
2954                     if (filtering) layer->setFiltering( false);
2955                 }
2956             }
2957         }
2958     }
```

Timing Fixed

- Time to take a screenshot and show the result screen inside an ImageView.
 - ~= 90ms.
- Fast enough for smooth user experience.
- Bug still present in Android 5.1.
- Patch:
 - <http://androidxref.com/filtering.patch>
 - My fault for not submitting for AOSP!

Emulator

- Unable to take screenshots
- Emulator uses OpenGL ES 1.0 when GPU acceleration is disabled.
- OpenGL ES 1.0 does not have support for framebuffers, which are required for taking screenshots
- Legacy screenshot code does not use framebuffers
- Workaround Patch:
 - [http:// androidxref.com/goldfish_screenshot.patch](http://androidxref.com/goldfish_screenshot.patch)

Emulator

- When GPU acceleration is enabled (-gpu on), OpenGL ES 1.1 is used.
- Workaround patch does not work for 1.1, but will generate a black screenshot instead of failing the operation.

Memory Consumption

- Capturing just a subregion of the screen requires two Bitmaps to be created.
 - Viewport/Projection is fixed.
 - Create a Bitmap of the entire screen then crop
- Easy to hit maximum heap size.
 - Bitmaps are allocated in the app heap to make it easier to debug.
 - `android:largeHeap="true"` helps but may not be enough

Android 5.0

- Support for custom projection and viewport
 - Allows to capture only a subregion of the screen.
 - Reduces the amount of memory required in the operation (Only one bitmap required, instead of two).

```
730     private static void screenshot(IBinder display, Surface consumer, Rect sourceCrop,
731                                   int width, int height, int minLayer, int maxLayer, boolean allLayers,
732                                   boolean useIdentityTransform) {

84 void GLES20RenderEngine::setViewportAndProjection(
85     size_t vpw, size_t vph, Rect sourceCrop, size_t hwh, bool yswap,
86     Transform::orientation_flags rotation) {
```



Rodrigo Chiossi
r.chiossi@androidxref.com
@rchiossi